



# Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search

Rémi Pautrat, Konstantinos Chatzilygeroudis, Jean-Baptiste Mouret

## ► To cite this version:

Rémi Pautrat, Konstantinos Chatzilygeroudis, Jean-Baptiste Mouret. Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search. ICRA 2018 - International Conference on Robotics and Automation, May 2018, Brisbane, Australia. hal-01768279

**HAL Id: hal-01768279**

**<https://inria.hal.science/hal-01768279>**

Submitted on 17 Apr 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Bayesian Optimization with Automatic Prior Selection for Data-Efficient Direct Policy Search

Rémi Pautrat, Konstantinos Chatzilygeroudis and Jean-Baptiste Mouret\*

**Abstract**—One of the most interesting features of Bayesian optimization for direct policy search is that it can leverage priors (e.g., from simulation or from previous tasks) to accelerate learning on a robot. In this paper, we are interested in situations for which several priors exist but we do not know in advance which one fits best the current situation. We tackle this problem by introducing a novel acquisition function, called Most Likely Expected Improvement (MLEI), that combines the likelihood of the priors and the expected improvement. We evaluate this new acquisition function on a transfer learning task for a 5-DOF planar arm and on a possibly damaged, 6-legged robot that has to learn to walk on flat ground and on stairs, with priors corresponding to different stairs and different kinds of damages. Our results show that MLEI effectively identifies and exploits the priors, even when there is no obvious match between the current situations and the priors.

## I. INTRODUCTION

Reinforcement Learning (RL) [1] could allow robots to adapt to new tasks (e.g., a new tool) and new contexts (e.g., a damage [2], [3]), but only if this adaptation happens in a few minutes: contrary to simulated worlds (e.g., games), where thousands (if not millions) of simulations can be evaluated, the number of trials in robotics hardware is limited by the energetic autonomy of the robot and the need to perform the task as soon as possible to be useful [4].

Among the different approaches to data-efficient RL, Bayesian Optimization (BO) is a promising approach because it can work with continuous action and state spaces, contrary to classic RL algorithms [5], and because it scales well with the dimension of the state space, contrary to model-based policy search algorithms (e.g., PILCO [6] or Black-DROPS [7]). For example, BO was successfully used to learn walking policies for a quadruped [8] and for a 2-legged compass walker [9].

BO was originally conceived as a black-box optimization algorithm for expensive functions [10], [11]. However, in robot learning, it is often possible to have some prior knowledge about the behavior of the system. For instance, a simulator of an intact robot can help to learn a policy on a damaged robot [2] or to guide the search algorithm to the most promising areas [12], [13]; or knowledge acquired when solving previous tasks can make it faster to solve a

new task (transfer learning) [14]. When BO is used for direct policy search, priors on the reward function can be added by using a non-constant mean function in the model, that is, by modeling the difference between the observations and the prior instead of modeling the observations directly [13], [2].

In this paper, we are interested in using BO when (1) several priors are available and, (2) we do not know beforehand which prior corresponds to the current context. A typical situation is a robot that knows how to solve a task in context A, B, and C (priors) and needs to learn to solve it in context D, while not knowing whether D is closer to A, B, or C. For some tasks, a perception system might recognize the right context [15], but in many others only the observations of the reward function can allow the robot to determine what prior is the most plausible. For instance, a walking robot could learn that a surface is slippery by observing that it matches the predictions that correspond to a prior for slippery floors, but it is often difficult to predict the slipperiness of a surface by only looking at it.

Our main insight is that we can compare two priors by computing the likelihood of the combination "prior + model" so that we can select the prior that matches the best the observations. Our second insight is that this prior selection can be elegantly incorporated as an acquisition function of a BO procedure, so that we select the next point to test by balancing between the expected improvement and the likelihood of the model used to compute the expected improvement. We demonstrate our approach on a simple simulated arm problem whose goal is to reach a target and on a simulated and physical 6-legged robot that faces different damage conditions and different environments.

## II. RELATED WORK

### A. Direct policy search in robotics

Direct policy search is a successful approach for RL in robotics because it scales well to high dimensional and continuous state-action spaces [5], [16]. Instead of trying to predict the expected returns of future events with value-function based learning as in TD learning [17], direct policy search algorithms look for the optimal parameters of parameterized policies. They essentially differ in the way the policy is updated, with techniques ranging from gradient estimation with finite differences [18] to more advanced optimization algorithms such as the Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [19], [20], Trust Region Policy Optimization (TRPO) [21] or Deep Deterministic Policy Gradient (DDPG) [22] algorithms. To make learning tractable, most of the successful experiments

\*Corresponding author: jean-baptiste.mouret@inria.fr

All authors have the following affiliations:

- Inria, Villers-lès-Nancy, F-54600, France

- CNRS, Loria, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France

- Université de Lorraine, Loria, UMR 7503, Vandœuvre-lès-Nancy, F-54500, France

This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (GA no. 637972, project "ResiBots") and the European Commission through the project H2020 AnDy (GA no. 731540).

rely on prior knowledge through demonstrations [5] and on low-dimensional policy representations (*e.g.*, dynamic movement primitives [20]): without such hand-designed priors, thousands of episodes are usually required [20], [5], [23].

Model-based policy search is an alternative to direct policy search that aims at improving the data-efficiency, that is, to minimize the number of required trials [6], [7]. To do so, model-based policy search algorithms choose the next policy by: (1) performing an episode on the robot, (2) learning a dynamical model of the system using the data acquired so far, and (3) optimizing the policy *according to the model* using a direct policy search algorithm. These algorithms scale well with the dimensionality of the policy (the number of parameters to optimize) because the policy optimization is performed on the model; but they are very sensitive to the dimension of the state-space because they need to learn to predict accurately the next state given the current one. Their difficulty to scale up makes it challenging to use them for systems that are more complex than basic control benchmarks (*e.g.*, cart-pole or simple manipulators) [5], [23].

### B. Bayesian optimization for RL

Instead of modeling the dynamics of the system, Bayesian Optimization (BO) directly models the reward function [24], [11]; it then leverages this model to predict the most promising set of parameters for the policy, that is, those that maximize the expected reward. After each episode, BO updates the model, which allows it to improve the predictions for the next iteration.

The core of Bayesian optimization is made of two main components: a model of the reward function, and an *acquisition function*, which uses the model to define the utility of each point of the search space. The vast majority of experiments with Bayesian optimization use Gaussian Processes (GP) [25] as a model. For the acquisition function, most of them use the Expected Improvement (EI), the Upper Confidence Bound (UCB) or the Probability of Improvement (PI) [24], [26]. Experimental results tend to show that EI can perform better on artificial objective functions than PI and UCB [26], but a recent experiment on gait learning on a physical robot suggested that UCB can outperform EI in real situations [9].

As a direct policy search approach, BO does not depend on the dimensionality of the state space, which makes it effective for learning policies for robots with complex dynamics (*e.g.*, locomotion, because of the non-linearity created by the contacts). For instance, Bayesian optimization was successfully used to learn policies for a quadruped robot [8], a quadcopter [27], a small biped “compass robot” [9], or a pocket-sized soft tensegrity robot [28]. In all of these cases, BO was at least an order of magnitude more data-efficient than competing methods. In a different domain, BO is also becoming one of the most successful approaches to tune the hyper-parameters of machine learning algorithms [11]: like in robotics, evaluating the quality of each set of parameters takes a long time.

### C. Priors for Gaussian processes

One benefit of using GP as a modeling method is that we can easily include prior knowledge about the data. The most common way is to select a particular *mean function*, which roughly corresponds to “what is the predicted value when there is no data?”.

Early work on Bayesian optimization for robotics focused on constant mean functions [8] (*i.e.*,  $\mu(x) = C$  where  $C$  is a user-defined constant). They noted that an overestimating mean function will make the real data appear mediocre, which leads to an excessive exploration, whereas an underestimating prior will lead to a greedy exploration since all the real observations will look promising [8].

More recent work proposed priors that come from simulators or simplified models, that is, non-constant priors. In particular, the “Intelligent Trial & Error” (IT&E) algorithm [2], [29] first creates a repertoire of about 15,000 high-performing policies and stores them in a low-dimensional map (*e.g.*, 6-dimensional whereas the policy space is 36-dimensional). When the robot needs to adapt, a Bayesian optimization algorithm searches for the best policy in the low-dimensional map and uses the reward stored in the map as the mean function of a GP. This algorithm allowed a 6-legged walking robot to adapt to several damage conditions (*e.g.*, a missing leg or a shortened leg) in less than 2 minutes (less than a dozen of trials), whereas it used a simulator of the intact robot to generate the prior. Instead of generating the prior first, it is also possible to choose between querying the simulator or the real robot [30] and add the point with a different “confidence level” (noise) depending on how they were obtained. Last, a recent article proposed to use a simulator to learn the kernel used in the GP, instead of using a simulator to define the mean function [31].

Priors from simulation were also successfully used in model learning with GPs: instead of learning the dynamical model of the robot from scratch, it is possible to learn a “residual model”, that is, the difference between the simulated and the real robot [13], [32], [33]. This approach was, for instance, successfully demonstrated with the PILCO algorithm for model-based policy search [12], [33] and when learning a model for optimal control [32].

While these contributions show that using well-chosen priors with GPs is a promising approach for data-efficient learning, all the previous algorithms assume that we know the “right” prior in advance. This is often a strong assumption because it means that the robot recognizes the current situation; this is also often a critical assumption because a misleading prior can substantially slow down the learning process. In the present paper, we relax this assumption by allowing the algorithm to choose the prior that is the most likely to help the learning process. For instance, we can have priors that correspond to different typical situations and let the algorithm choose automatically the most relevant one (and ignore the misleading ones).

### III. COMBINING LIKELIHOOD AND EXPECTED IMPROVEMENT

Like in most BO implementations, we model the objective function  $F(\mathbf{x})$  to be maximized over the space  $\mathcal{X}$  by a Gaussian process  $f(\mathbf{x})$  with a mean function  $m(\cdot)$  and a covariance function  $\kappa(\cdot, \cdot)$ :

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$$

Let us assume that we already made  $t$  observations on the points  $\mathbf{x}_1, \dots, \mathbf{x}_t \in \mathcal{X}$  (abridged as  $\mathbf{x}_{1..t}$ ) that are summed up in the vector  $\mathbf{F}(\mathbf{x}_{1..t}) = (F(\mathbf{x}_1), \dots, F(\mathbf{x}_t))$ , and that we fixed a noise parameter  $\sigma_n$ . The GP for a new point  $\mathbf{x} \in \mathcal{X}$  is computed using a kernel function  $\kappa(\mathbf{x}, \mathbf{y})$ , a kernel vector  $\mathbf{k}$ , and a kernel matrix  $\mathbf{K}$  [25]:

$$P(f(\mathbf{x}) \mid \mathbf{x}_{1..t}) \sim \mathcal{N}(\mu_t(\mathbf{x}), \sigma_t^2(\mathbf{x})), \text{ where : } \quad (1)$$

$$\mu_t(\mathbf{x}) = \mathbf{k}^T \mathbf{K}^{-1} \mathbf{F}(\mathbf{x}_{1..t}) \quad (2)$$

$$\sigma_t^2(\mathbf{x}) = \kappa(\mathbf{x}, \mathbf{x}) - \mathbf{k}^T \mathbf{K}^{-1} \mathbf{k} \quad (3)$$

$$\mathbf{K} = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \dots & \kappa(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_t, \mathbf{x}_1) & \dots & \kappa(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} + \sigma_n^2 \mathbf{I} \quad (4)$$

$$\mathbf{k} = [\kappa(\mathbf{x}, \mathbf{x}_1) \quad \dots \quad \kappa(\mathbf{x}, \mathbf{x}_t)]^T \quad (5)$$

In many situations, some prior knowledge on the objective function is available before starting the optimization. In that case, we can write this information with a prior function  $\mathcal{P}$  and update the equations of the GP accordingly [2]:

$$\mu_t(\mathbf{x}) = \mathcal{P}(\mathbf{x}) + \mathbf{k}^T \mathbf{K}^{-1} (\mathbf{F}(\mathbf{x}_{1..t}) - \mathcal{P}(\mathbf{x}_{1..t})) \quad (6)$$

The next point  $\mathbf{x}$  where the objective function should be evaluated is found by maximizing an *acquisition function*, that is, a function that leverages the model (both the variance and the mean) to predict the most promising point. A function that is often used for this is the Expected Improvement (EI) [34], [11]:

$$\begin{aligned} \text{EI}(\mathbf{x}) &= \mathbb{E}(\text{I}(\mathbf{x})) = \mathbb{E}(\max(0, f(\mathbf{x}) - M_t)) \\ &= \begin{cases} (\mu_t(\mathbf{x}) - M_t)\Phi(Z) + \sigma_t(\mathbf{x})\phi(Z) & \text{if } \sigma_t(\mathbf{x}) \neq 0 \\ 0 & \text{if } \sigma_t(\mathbf{x}) = 0 \end{cases} \quad (7) \end{aligned}$$

where  $M_t = \max_{i=1..t} F(\mathbf{x}_i)$  is the best value observed at time  $t$ ,  $Z = \frac{\mu_t(\mathbf{x}) - M_t}{\sigma_t(\mathbf{x})}$ ,  $\Phi$  and  $\phi$  are respectively the cumulative and probability density functions of the standard normal distribution.

Choosing the best prior can be seen as a problem of model selection (since the prior is part of the model), which is effectively achieved by comparing the likelihood of alternative models [25]:

$$\begin{aligned} P(\mathbf{f}(\mathbf{x}_{1..t}) \mid \mathbf{x}_{1..t}, \mathcal{P}(\mathbf{x}_{1..t})) &= \\ \frac{1}{\sqrt{(2\pi)^t |K|}} \exp \left( -\frac{1}{2} (\mathbf{F}(\mathbf{x}_{1..t}) - \mathcal{P}(\mathbf{x}_{1..t}))^T \mathbf{K}^{-1} (\mathbf{F}(\mathbf{x}_{1..t}) - \mathcal{P}(\mathbf{x}_{1..t})) \right) \quad (8) \end{aligned}$$

Intuitively, we could select the prior that corresponds to the best likelihood, then compute the expected improvement

for this model. However, we would risk to select an “over-pessimistic” prior at the beginning of the optimization, because the first observations (which are often random points) are likely to be low-performing — if random points were likely to be high-performing, there would be no need for learning. In essence, if we have not yet observed any high-performing solutions, then the likeliest prior is a prior for which every solution is low-performing.

We therefore need to balance between the likelihood of the prior and the potential for high-performing solutions. In other words, a good expected improvement according to an unlikely model should be ignored; conversely, a likely model with a low expected improvement might be too pessimistic (“nothing works”) and not helpful. A model that is “likely enough” and lets us expect some good improvement might be the most helpful to find the maximum of  $F$ .

Let us assume that the objective function only takes discrete values, in which case the likelihood is a probability. Considering  $t$  observations  $F(\mathbf{x}_1), \dots, F(\mathbf{x}_t)$ , we introduce the indicator function  $\mathbb{1}_{f(\mathbf{x}_1)=F(\mathbf{x}_1), \dots, f(\mathbf{x}_t)=F(\mathbf{x}_t)}$  which equals to 1 when the predictions match exactly the observations, and we define the Expected Improvement for a prior  $\mathcal{P}$ :

$$\begin{aligned} \text{EIP}(\mathbf{x}, \mathcal{P}) &= \mathbb{E}(\text{I}(\mathbf{x}) \times \mathbb{1}_{f(\mathbf{x}_1)=F(\mathbf{x}_1), \dots, f(\mathbf{x}_t)=F(\mathbf{x}_t)}) \quad (9) \\ &= \mathbb{E}(\max(0, f(\mathbf{x}) - M_t) \times \mathbb{1}_{f(\mathbf{x}_1)=F(\mathbf{x}_1), \dots, f(\mathbf{x}_t)=F(\mathbf{x}_t)}) \end{aligned}$$

But as the predicted value  $f(\mathbf{x}) \sim \mathcal{N}(\mu_t(\mathbf{x}), \sigma_t^2(\mathbf{x}))$  only depends on the samples  $\mathbf{x}_1, \dots, \mathbf{x}_t$ , the observations  $\mathbf{F}(\mathbf{x}_{1..t})$  and the deterministic function  $\mathcal{P}$ , it is independent of the original distribution  $\mathbf{f}(\mathbf{x}_{1..t}) \sim \mathcal{N}(\mathcal{P}(\mathbf{x}_{1..t}), \mathbf{K})$ . Thus the two factors inside the expectation are two independent variables and can be split:

$$\begin{aligned} \text{EIP}(\mathbf{x}, \mathcal{P}) &= \mathbb{E}(\max(0, f(\mathbf{x}) - M_t)) \\ &\quad \times \mathbb{E}(\mathbb{1}_{f(\mathbf{x}_1)=F(\mathbf{x}_1), \dots, f(\mathbf{x}_t)=F(\mathbf{x}_t)}) \quad (10) \\ &= \text{EI}(\mathbf{x}) \times P(\mathbf{f}(\mathbf{x}_{1..t}) = \mathbf{F}(\mathbf{x}_{1..t}) \mid \mathbf{x}_{1..t}, \mathcal{P}) \end{aligned}$$

This new function can be extended afterwards to the case where  $F$  takes continuous values: the likelihood becomes a density probability function, but the EIP can still be defined as the product of the expected improvement with the likelihood:

$$\text{EIP}(\mathbf{x}, \mathcal{P}) = \text{EI}(\mathbf{x}) \times P(\mathbf{f}(\mathbf{x}_{1..t}) \mid \mathbf{x}_{1..t}, \mathcal{P}(\mathbf{x}_{1..t})) \quad (11)$$

When we have  $m$  priors  $\mathcal{P}_1, \dots, \mathcal{P}_m$ , the Most Likely Expected Improvement (MLEI) acquisition function can then be defined as:

$$\text{MLEI}(\mathbf{x}, \mathcal{P}_1, \dots, \mathcal{P}_m) = \max_{p \in \mathcal{P}_1, \dots, \mathcal{P}_m} \text{EIP}(\mathbf{x}, p) \quad (12)$$

The MLEI acquisition function can be used like any other acquisition function in the BO algorithm. Please note that the likelihood has to be evaluated only once for each model (that is, once for each prior), and not for every point  $\mathbf{x}$  (see Algo. 1). We use the C++-11 Limbo library for the BO implementation [35].

---

**Procedure 1** Bayesian Optimization with MLEI

---

```

1: procedure BOMULTIPLEPRIORS
2: Input:  $m$  priors  $\mathcal{P}_1, \dots, \mathcal{P}_m$ , an objective function  $F$ 
3: Output: An approximation of the maximum of  $F$ 
4: Initialize  $m$  Gaussian processes  $f_1, \dots, f_m$  with the
    $m$  priors and the kernel function  $\kappa$ :
5:  $\forall i \in \{1, \dots, m\}, f_i(\mathbf{x}) \sim \mathcal{N}(\mathcal{P}_i(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}'))$ 
6:  $t \leftarrow 1$ 
7: while  $t \leq \text{maxIterations}$  do
8:   for  $i = 1..m$  do
9:      $l \leftarrow \text{computeLikelihood}(f_i, \mathbf{x}_1, \dots, \mathbf{x}_{t-1})$ 
10:     $\mathbf{s}_i \leftarrow \text{argmax}_{\mathbf{x} \in \mathcal{X}} \text{EI}(\mathbf{x})$ 
         $= \text{argmax}_{\mathbf{x} \in \mathcal{X}} (\mathbb{E}(\max(0, f_i(\mathbf{x}) - M_t)))$ 
11:     $\text{EIP}(\mathbf{s}_i, \mathcal{P}_i) \leftarrow l \times \text{EI}(\mathbf{s}_i)$ 
12:     $\mathbf{x}_t, p_t \leftarrow \text{argmax}_{i=1..m} \text{EIP}(\mathbf{s}_i, \mathcal{P}_i)$ 
13:    Evaluate  $F(\mathbf{x}_t)$  on the robot
14:    Update the  $m$  Gaussian processes with the new
    observation  $F(\mathbf{x}_t)$ 
15:     $t \leftarrow t + 1$ 
return  $\max_{t=1..maxIterations} F(\mathbf{x}_t)$ 

```

---

#### IV. EXPERIMENTAL RESULTS

##### A. Robotic arm experiment (transfer learning)

We first evaluate the MLEI acquisition function with a kinematic simulation of a planar robotic arm that has to reach a target point with its end effector (Fig.1(a)). The arm has 5 Degrees of Freedom (DOFs) and each link measures 1m. The reward function is the distance between the end effector and the target point  $[3, 3]$  (we use a negative distance because our implementation of BO maximizes the reward). The robot is position controlled and the joints can take positions in  $[-\pi; \pi]$ . The policy is parametrized by the 5 target angles.

We pre-defined 10 priors (*i.e.*, 10 mean functions  $\mathcal{P}(\mathbf{x})$ ) using the function  $\text{FWD}(\mathbf{x})$ , which gives the position of the end effector given the angular positions  $\mathbf{x}$  and the forward kinematics:

- the null prior:  $\mathcal{P}_1(\mathbf{x}) = 0$  (*i.e.*, the traditional BO algorithm);
- $\mathcal{P}_2(\mathbf{x}) = -\|\text{FWD}(\mathbf{x}) - \mathbf{y}_2\|$ ,  $\mathbf{y}_2 = [3.6, 3.3]$  (this corresponds to a good prior since the point  $\mathbf{y}_2$  is close to the actual target);
- $\mathcal{P}_3(\mathbf{x}) = -\|\text{FWD}(\mathbf{x}) - \mathbf{y}_3\|$ ,  $\mathbf{y}_3 = [2, 2]$  (fair prior);
- $\mathcal{P}_4(\mathbf{x}) = -\|\text{FWD}(\mathbf{x}) - \mathbf{y}_4\|$ ,  $\mathbf{y}_4 = [0, 0]$  (not so good prior);
- $\mathcal{P}_5(\mathbf{x}) = -\|\text{FWD}(\mathbf{x}) - \mathbf{y}_5\|$ ,  $\mathbf{y}_5 = [-3, -3]$  (bad prior);
- $\mathcal{P}_{6,\dots,10}(\mathbf{x}) = -\|\text{FWD}(\mathbf{x}) - \mathbf{y}_{5+i}\|$  where  $i = 1, \dots, 5$  and  $\mathbf{y}_{5+i}$  is randomly chosen in  $[-3, 3]$  (this is done once before all the experiments; see Fig. 1(a) for the target points used in the experiments).

None of these priors corresponds to the right target, but for instance, if the second prior is selected, then the robot “knows” how to reach the target at  $[3.6, 3.3]$ . This setup can be seen as a simple transfer learning example: (1) the robot knows how to reach some targets, that is, how to solve some tasks, and (2) the robot needs to learn how to reach a novel target given the knowledge of previous targets.

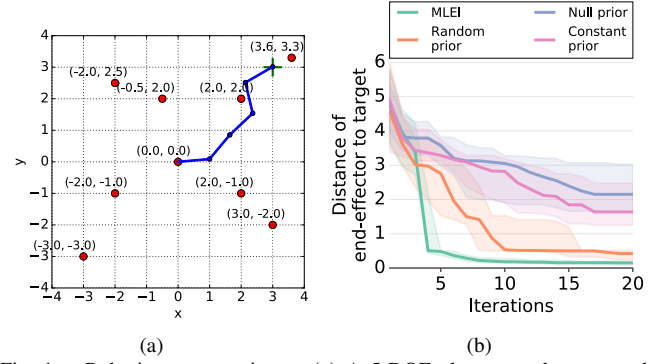


Fig. 1. Robotic arm experiment. (a) A 5-DOF planar arm has to touch a given target with its end-effector. The red circles correspond to the target points of the available priors, whereas the green cross indicates the actual target. A solution found by MLEI is shown. (b) Comparison of the MLEI acquisition function with EI without prior (traditional BO), EI with a constant prior mean function and EI with a random selection of priors. 30 replicates of the experiment have been carried out, each one of them consisting in 20 iterations of BO (including 3 initial random trials).

The optimization is initialized by three random trials of the robotic arm and then BO is used for 17 iterations to select the next move of the arm (for a total of 20 episodes on the robot). We compare four different variants of BO:

- EI with null prior: standard BO using EI without prior (the mean function is equal to 0 — this is an optimistic prior [36]);
- EI with constant prior: BO using EI with constant prior (the mean function is equal to  $-7$  — this is a pessimistic prior [36]);
- EI with a prior randomly selected among the available priors at each iteration of BO;
- MLEI with automatic selection of priors at each iteration of BO.

We replicated each experiment 30 times to gather statistics.

The results show that MLEI finds a policy that reaches the target (distance to the end effector inferior to 15cm) after 7 – 8 episodes, whereas the EI with random selection of priors and the EI with no prior need more than 20 (Fig. 1(b)). Overall, MLEI clearly outperforms the three baselines.

##### B. 6-legged robot experiment (adaptation to new environments and to damage)

We then evaluate the MLEI acquisition function in a similar context as in Cully *et al.* [2]: a 6-legged robot is either damaged in an unknown way or introduced to an unknown environment and BO is used to find an alternative walking gait that works in spite of the unforeseen situation. However, while Cully *et al.* used a single prior (walking on a flat surface with an intact robot), we introduce many other priors that correspond either to potential damages (*e.g.*, a missing leg) or to different terrains (*e.g.*, stairs). We test the learning algorithm with priors corresponding to the actual situation, but also in situations that are not fully covered by any prior.

1) *Robot and policy:* The robot has identical legs with 3 DOFs per leg (Fig. 2(b)). One DOF ( $\theta_1$ ) controls the horizontal movements of the leg (from back to front) whereas the two others ( $\theta_2$  and  $\theta_3$ ) control the elevation of the leg. Each one of these DOFs is controlled by an open-loop

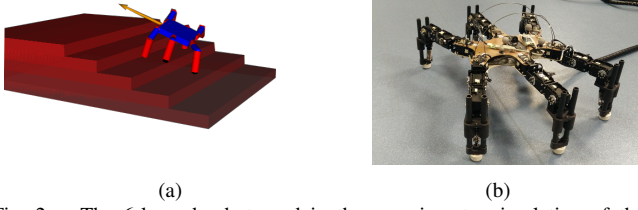


Fig. 2. The 6-legged robot used in the experiments: simulation of the hexapod on stairs (a) and real robot (b).

oscillator defined with 3 parameters [2]: an amplitude, a phase, and a duty cycle (proportion of time in which the angle is in an extreme position). The second vertical angle  $\theta_3$  is constrained to take values between  $-\theta_2$  and  $-\theta_2 + \frac{\pi}{4}$ , so that the inferior member (the "tibia") remains vertical or at most at an angle of  $\frac{\pi}{4}$  with the vertical line. Thus, the whole gait of the robot can be defined with  $6 \times 3 \times 3 = 54$  parameters. All simulations<sup>1</sup> of the robot are performed with the Dynamic Animation and Robotics Toolkit (DART)<sup>2</sup> in a world with gravity where the simulated robot is similar to the intact, physical hexapod.

2) *Reward function*: In all the experiments, the reward function is the distance covered by the 6-legged robot in a virtual corridor with a width of 1m (the width of the robot is about 40cm). As soon as the robot gets out of the corridor, the evaluation is stopped; it is also stopped after 10s if the robot stays in the corridor. Compared to more traditional reward functions, for instance the distance covered in 10 seconds, our reward function encourages more the robot to follow a straight line, even if it means that the gait is slower. Similar results were however obtained with the average walking speed as a reward.

3) *Prior generation*: All the priors are 6-dimensional behavior-reward maps computed for a simulated 6-legged robot in different environments or with the damaged robot (e.g., with a missing leg). These behavior-reward maps are created beforehand using the MAP-Elites algorithm [2], [37], which is a recent evolutionary algorithm designed to generate thousands of different high-performing control policies<sup>3</sup>.

MAP-Elites assumes that we can define a low-dimensional *behavior descriptor* for each policy, that is, a low-dimensional vector that captures the main difference between two behaviors. Given a  $n$ -dimensional behavior space, MAP-Elites defines a  $n$ -dimensional grid divided in cells, and attempts to fill each of the cells with high-quality solutions. To do so, it starts with  $G$  random policy parameters, simulates the robot with these parameters, and records both the position of the robot in the behavior space and the performance. If the cell is free, then the algorithm stores the policy parameters in that cell; if it is already occupied, then the algorithm compares the reward values and keeps only the

best parameter vector. Once this initialization is done, MAP-Elites iterates a simple loop: (1) randomly select one of the occupied cells, (2) add a random variation to the parameter vector, (3) simulate the behavior, (4) insert the new parameter vector into the grid if it performs better or ends up in an empty cell (discard the new parameter vector otherwise).

MAP-Elites can be straightforwardly parallelized and can run on large clusters before deploying the robot. So far, it has been successfully used to create behaviors for legged robots [2], [3], wheeled robots [39], [38], designs for airfoils [40], morphologies of walking "soft robots" [37], and adversarial images for deep neural networks [41]. MAP-Elites has also been extended to effectively handle tasks with spaces of arbitrary dimensionality [42].

We use one of the behavior descriptors proposed in Cully *et al.* [2]: the body orientation, which captures how often the body of the robot is tilted in each direction<sup>4</sup>. More formally, simulating each policy leads to a 6-dimensional vector that contains the proportion of time that the body of the robot has a positive and negative pitch, yaw and roll:

$$BOF = \begin{bmatrix} \frac{1}{K} \sum_{k=1}^K \mathbb{1}_{\Theta(k) > 0.005\pi} \\ \frac{1}{K} \sum_{k=1}^K \mathbb{1}_{\Theta(k) < -0.005\pi} \\ \frac{1}{K} \sum_{k=1}^K \mathbb{1}_{\Psi(k) > 0.005\pi} \\ \frac{1}{K} \sum_{k=1}^K \mathbb{1}_{\Psi(k) < -0.005\pi} \\ \frac{1}{K} \sum_{k=1}^K \mathbb{1}_{\Phi(k) > 0.005\pi} \\ \frac{1}{K} \sum_{k=1}^K \mathbb{1}_{\Phi(k) < -0.005\pi} \end{bmatrix} \quad (13)$$

where the duration of the gait of the robot is divided in  $K$  intervals of 15 ms,  $\Theta$ ,  $\Psi$  and  $\Phi$  are the pitch, roll and yaw of the torso of the robot, respectively,  $\mathbb{1}$  is the indicator function which returns 1 if and only if its argument is true, and angles between  $\pm 0.005\pi$  are ignored.

This quantity is rounded so that it can only take values in  $\{0, 0.2, 0.4, 0.6, 0.8\}$  and so the set of all the body orientation factors is finite and contains  $5^6 = 15625$  elements that can be organized in a map.

For the purpose of the experiments, 15 behavior-performance maps have been created for each of the possible environments (priors). Each one of these maps was created with a run of the MAP-Elites algorithm for 24 hours on a 16-core Xeon computer. We used the Sferes C++ library [43].

The kernel chosen for the GP is the Squared Exponential Kernel:  $k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \mathbf{M}(\mathbf{x} - \mathbf{x}')\right)$  where  $\mathbf{M} = \text{diag}(l_1^{-2}, \dots, l_D^{-2})$  is the characteristic length scales (here  $D = 6$ ) [24] [25]. Initially,  $\sigma = 1$  and  $\forall i \in \{1, \dots, D\}, l_i = 1$  and  $\sigma_n = 0.00001$ . The hyperparameters of the kernel are optimized through Resilient backPROPagation (RPROP) [44], with 300 iterations.

4) *Experiment 1 — Adaptation to stairs in simulation*: In our first set of experiments, the intact robot needs to adapt to unknown environments. We generated 15 behavior-performance maps (i.e., 15 priors for the GP) for each of the four following environments:

- flat ground;

<sup>4</sup>Similar results were obtained with other behavioral descriptors.

<sup>1</sup>[https://github.com/resibots/robot\\_dart](https://github.com/resibots/robot_dart)

<sup>2</sup><http://dartsim.github.io/>

<sup>3</sup>This new family of search algorithm is called "illumination algorithms" [37] or "quality diversity algorithms" [38].



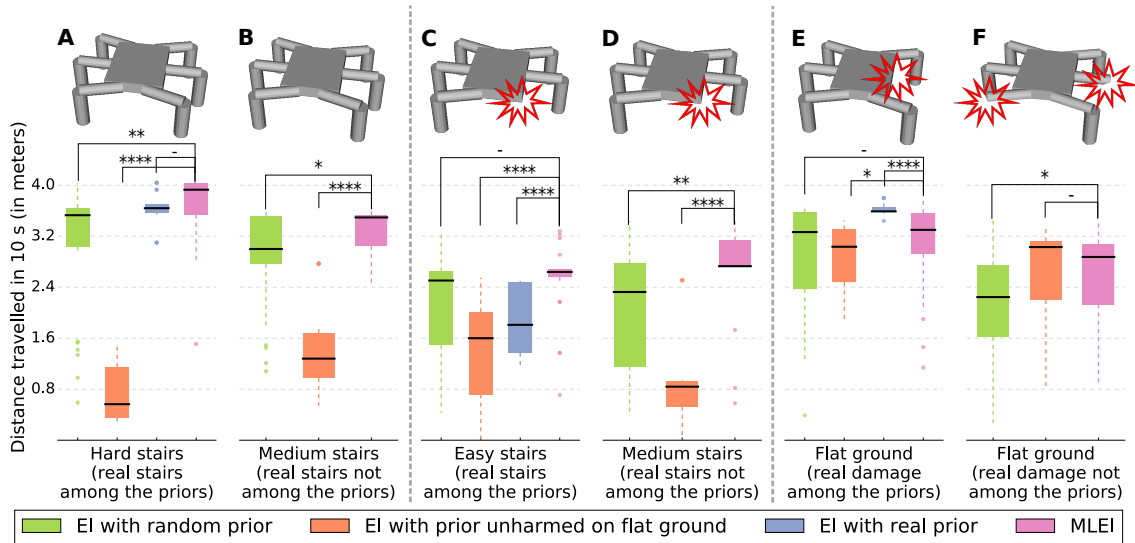


Fig. 3. Comparison in simulation of MLEI with other acquisition functions and choices of prior (EI with a randomly selected prior, EI with a prior generated on an unharmed robot on flat ground and EI with the prior corresponding to the real stairs or damage) on the 6-legged robot learning to climb stairs and/or to recover from damages after 5 iterations of BO and with 30 replicates of each experiment. (A) and (B): the robot is on unknown stairs with no damage and the real stairs can be among the priors (A) or not (B). (C) and (D): the robot is on unknown stairs with unknown damages and the priors are only on stairs not on damages (the actual stairs can be among the priors (C) or not (D)). (E) and (F): the robot is on flat ground with unknown damages and the real damage can be among the priors (E) or not (F). The number of stars indicates that the p-value, obtained using the Mann-Whitney-Wilcoxon test, is below 0.0001, 0.001, 0.01 and 0.05 respectively.

- easy stairs (steps with height: 4cm, width: 1.2m, depth: 50cm);
- medium stairs (steps with height: 5cm, width: 1.2m, depth: 20cm);
- hard stairs (steps with height: 7.5cm, width: 1.2m, depth: 25cm).

We compare the following acquisition functions for BO:

- EI with a single prior coming from a simulated robot on flat ground – this corresponds to the original IT&E experiments [2];
- EI with a single prior, randomly chosen among the available priors at each iteration.
- EI with a single prior coming from a simulated robot on the actual stairs (when available) – this corresponds to the ideal case, in which we know the right prior;
- MLEI with a prior selected at each iteration among the available priors (flat ground, easy, medium and hard stairs).

For the MLEI and EI with random priors experiments, we randomly choose 5 priors (*i.e.*, 5 maps) for each possible environment, leading to a unique set of priors for each MLEI experiment and for each experiment with randomly chosen priors. Please note that several priors correspond to the same situation, which is interesting because some maps might be of higher-quality than others, even if they have been generated with the same environment.

We test two situations:

- 1) adaptation to hard stairs when the hard stairs are part of the priors given to MLEI (and to random selection) —  $5 \times 4 = 20$  priors to select from;
- 2) adaptation to medium stairs, with the medium stairs removed from the priors given to MLEI (and to random selection) —  $5 \times 3 = 15$  priors to select from.

In these two situations, the robot is the same in the prior and in the adaptation experiment (there is no “reality gap”).

The results (Fig. 3A-B) show that MLEI allows the robot to learn high-performing gaits for the stairs, even when the stairs used for the learning experiments are not present in the set of priors (Fig. 3B): when the right prior is accessible, MLEI finds it; when it is not accessible, it can still leverage other priors and use BO to find a good behavior while using other priors. In the two tested cases, MLEI clearly outperforms the random selection of priors and the method using the flat ground prior, which means that MLEI selects priors correctly and that these priors help the learning process. Surprisingly, MLEI also outperforms the EI with a “perfect” prior (Fig. 3A): this is because MLEI has access to 5 priors for the hard stairs (in addition to the 15 other priors) and therefore can select the best of them, whereas each EI experiment has access to a single prior for the considered stairs (and the best controller for each map is different). The relatively good performance of the random selection of priors is likely to stem from the fact that this algorithm has access to a much higher diversity of behaviors than EI with flat ground as a prior (that is, to the original IT&E), which makes it more likely to find a behavior that works in the tested situation.

**5) Experiment 2 — Adaptation to stairs and damages in simulation:** In this second experiment, we evaluate if the robot can adapt to unforeseen damage conditions, with and without stairs, with and without priors about the damage conditions. For each of the 6 legs removed, we generated 15 priors with MAP-Elites (with a robot on flat ground), leading to  $(6 + 1) \times 15 = 105$  priors (6 damage conditions + intact robot). Like in the previous experiments, the set of available priors is made of 5 random maps (out of the 15 generated priors) for each situation.

We compare the same methods as before in four situations

that cover different combinations of environmental and body-related priors:

- 1.a adaptation to damage with priors about stairs (no prior about damage), and when the actual stairs are among the priors — 20 priors to select from;
- 1.b adaptation to damage with priors about stairs (no prior about damage), and when the actual stairs are not among the priors — 15 priors to select from;
- 2.a adaptation to damage with priors about the damage conditions, on flat ground, when the actual damage (left middle leg removed) is among the priors —  $7 \times 5 = 35$  priors to select from;
- 2.b adaptation to damage with priors about the damage conditions, on flat ground, when the actual damage (front right leg and middle left leg shortened) is not among the priors —  $7 \times 5 = 35$  priors to select from.

The results (Fig. 3C-F and supplementary video<sup>5</sup>) show that MLEI can find compensatory gaits on stairs while using priors computed with the intact robot. When the real stairs are among the priors (Fig. 3C), MLEI outperforms the EI with the right stairs because (1) since the robot is damaged, the most helpful prior is not always the prior that corresponds to the correct stairs (*e.g.*, the prior that corresponds to the hard stairs might be more conservative and be more helpful when the robot is damaged); (2) like before, MLEI has access to more priors, which makes it more likely to have a policy in one of the map that can compensate for the damage.

When the actual stairs are not in the priors, MLEI still outperforms the two other approaches (Fig. 3D). MLEI can also take advantage of priors about the damage condition whether the damage is included in the priors or not (Fig. 3E-F): when the actual damage conditions is among the priors, MLEI leads to higher-performing solutions than EI with the intact robot as a prior; when the damage condition is not among the prior, MLEI performs the same as EI with the intact robot as a prior. These results are consistent with [2], which shows that an intact robot can be an effective prior to adapt to damage.

6) **Experiment 3 — Adaptation to damage with a physical robot:** In this experiment, we use  $(6 + 1) \times 15 = 105$  priors for damage conditions to allow a physical 6-legged robot to adapt. As the simulation is not perfect, the learning algorithm has to compensate for both the “reality gap” and the damage. The robot is tracked with an external motion capture system (Optitrack) and we use 10 episodes of 10 seconds. Like before, we consider two situations: when the damage is among the priors, and when it is not. We replicate each experiment 5 times.

Like in simulation, MLEI takes advantage of the priors to find higher-performing gaits than when a single prior is used (Fig. 4 and supplementary video<sup>5</sup>). When one of the priors correspond to the actual damage condition (Fig. 4(a)), MLEI clearly outperforms EI with a single prior and finds high-performing gaits in less than 10 episodes; MLEI also finds better gaits than EI when the actual damage condition

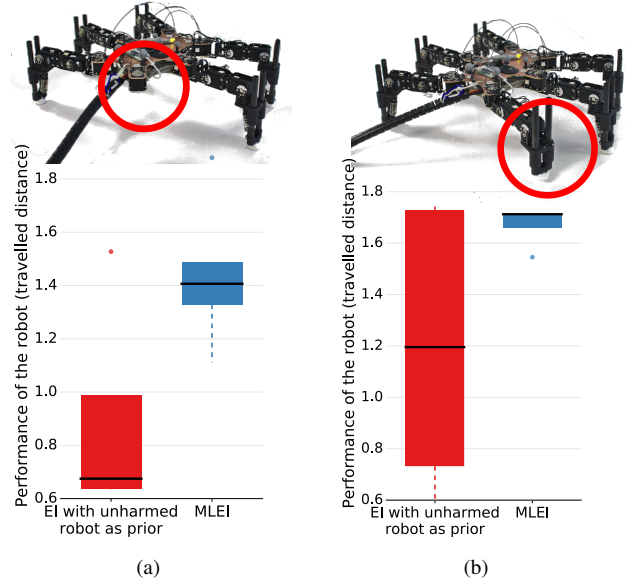


Fig. 4. Comparison of MLEI with the standard EI with a single prior coming from a simulated undamaged robot. This real experiment was carried out on the physical damaged 6-legged robot walking on flat ground after 10 iterations of BO and with 5 replicates of the experiment. Damage used: (a) missing rear leg (damage present among the available priors), (b) shortened rear leg (damage not present among the available priors).

is not among the priors (Fig. 4(b)), which is likely to come from the fact that MLEI can “take inspiration” from other priors to compensate for the damage (like in the previous task, this corresponds to a form of transfer learning).

## V. CONCLUSION AND DISCUSSION

Well-chosen priors can guide BO to find a high-performing solution [8], [2] while not constraining the search to a few pre-designed solutions. However, learning algorithms are most useful when the robot or the environment are partially known, therefore it is often challenging to design a single prior that would help BO in all the possible situations. The Most Likely Expected Improvement (MLEI) allows us to relax this assumption by making BO capable of selecting the most useful prior and ignore all the others. It therefore makes it possible for BO to benefit from the faster convergence speed given by the priors, while not assuming much about the robot or the environment.

In this paper, we demonstrated that our new acquisition function leads to a powerful adaptation algorithm in two systems, a planar manipulator and a 6-legged robot. In the latter case, the robot was capable of discovering compensatory behaviors in a dozen of trials when damaged — even with priors that correspond to the intact robot — and when it faced unknown stairs — even without any prior for the actual stairs. Overall, MLEI substantially increases the potential uses of priors in BO because we can often “guess” what could be useful to the robot, but we cannot be sure in advance if a given prior will be useful in the future.

Even the best classification system based on perception (which context is recognized by the robot?) [15] is prone to errors in real situations (*e.g.*, steps hidden by high grass). By contrast, the automatic selection of priors that we introduced

<sup>5</sup>Also at: <https://youtu.be/xo8mUIZTvNE>



here is based on the direct observation of the rewards: the robot discovers what works and what does not, it does not attempt to know *why* some behaviors work and some do not. This approach fits well the theory of “embodied cognition” [45], [46] which suggests that robots do not need an explicit representation of the world to act. A classic “sense-plan-act” architecture would assume the existence of an accurate model of the world to act; at the other end of the spectrum, most learning algorithms aim at assuming as little knowledge as possible about the robot or the environment. BO with automatic selection of prior can be an effective middle ground in which prior knowledge or a perception system can guide a direct policy search that can, if needed, ignore all previous knowledge and still find an effective way to act.

#### APPENDIX

Code of the experiments: [https://github.com/resibots/pautrat\\_2018\\_mlei](https://github.com/resibots/pautrat_2018_mlei)

#### REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998.
- [2] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, “Robots that can adapt like animals,” *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.
- [3] K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret, “Reset-free Trial-and-Error Learning for Robot Damage Recovery,” *arXiv:1610.04213*, 2016.
- [4] J.-B. Mouret, “Micro-data learning: The other end of the spectrum,” *ERCIM News*, 2016.
- [5] M. P. Deisenroth, G. Neumann, and J. Peters, “A survey on policy search for robotics,” *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.
- [6] M. Deisenroth, D. Fox, and C. Rasmussen, “Gaussian processes for data-efficient learning in robotics and control,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2015.
- [7] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, “Black-Box Data-efficient Policy Search for Robotics,” in *Proc. of IROS*, 2017.
- [8] M. B. D. Lizotte, T. Wang and D. Schuurmans, “Automatic Gait Optimization with Gaussian Process Regression,” in *Proc. of IJCAI*, 2007.
- [9] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, “Bayesian optimization for learning gaits under uncertainty,” *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1–2, pp. 5–23, 2016.
- [10] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.
- [11] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, “Taking the human out of the loop: A review of bayesian optimization,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.
- [12] M. Cutler and J. P. How, “Efficient reinforcement learning for robots using informative simulated priors,” in *Proc. of ICRA*, 2015.
- [13] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, “Gaussian processes and reinforcement learning for identification and control of an autonomous blimp,” in *Proc. of ICRA*, 2007.
- [14] M. E. Taylor and P. Stone, “Transfer learning for reinforcement learning domains: A survey,” *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1633–1685, 2009.
- [15] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard, “Learning predictive terrain models for legged robot locomotion,” in *Proc. of IROS*, 2008.
- [16] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “A brief survey of deep reinforcement learning,” *arXiv preprint arXiv:1708.05866*, 2017.
- [17] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Machine Learning*, vol. 3, no. 1, pp. 9–44, Aug 1988.
- [18] N. Kohl and P. Stone, “Policy gradient reinforcement learning for fast quadrupedal locomotion,” in *Proc. of ICRA*, 2004.
- [19] N. Hansen and A. Ostermeier, “Completely derandomized self adaptation in evolution strategies,” *Evolutionary Computation*, pp. 159–195, 2001.
- [20] F. Stulp and O. Sigaud, “Robot skill learning: From reinforcement learning to evolution strategies,” *Paladyn, Journal of Behavioral Robotics*, vol. 4, no. 1, pp. 49–61, 2013.
- [21] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” in *Proc. of ICML*, 2015.
- [22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *arXiv preprint arXiv:1509.02971*, 2015.
- [23] A. S. Polydoros and L. Nalpantidis, “Survey of model-based reinforcement learning: Applications on robotics,” *Journal of Intelligent & Robotic Systems*, pp. 1–21, 2017.
- [24] V. M. C. E. Brochu and N. de Freitas, “A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” *CoRR*, vol. abs/1012.2599, 2010.
- [25] E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.
- [26] P. Hennig and C. J. Schuler, “Entropy search for information-efficient global optimization,” *Journal of Machine Learning Research*, vol. 13, 2011.
- [27] F. Berkenkamp, A. P. Schoellig, and A. Krause, “Safe controller optimization for quadrotors with gaussian processes,” in *Proc. of ICRA*, 2016.
- [28] J. Rieffel and J.-B. Mouret, “Soft tensegrity robots,” *arXiv preprint arXiv:1702.03258*, 2017.
- [29] V. Papaspyros, K. Chatzilygeroudis, V. Vassiliades, and J.-B. Mouret, “Safety-aware robot damage recovery using constrained bayesian optimization and simulated priors,” in *BayesOpt ’16 Workshop at NIPS*, 2016.
- [30] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, “Virtual vs. Real: Trading Off Simulations and Physical Experiments in Reinforcement Learning with Bayesian Optimization,” in *Proc. of ICRA*, 2017.
- [31] R. Antonova, A. Rai, and C. G. Atkeson, “Deep Kernels for Optimizing Locomotion Controllers,” in *Proc. of CoRL*, 2017.
- [32] G. Lee, S. S. Srinivasa, and M. T. Mason, “GP-ILQG: Data-driven Robust Optimal Control for Uncertain Nonlinear Dynamical Systems,” *arXiv preprint arXiv:1705.05344*, 2017.
- [33] M. Saveriano, Y. Yin, P. Falco, and D. Lee, “Data-Efficient Control Policy Search using Residual Dynamics Learning,” in *Proc. of IROS*, 2017.
- [34] I. O. R. Warren B. Powell, *Optimal Learning*. Wiley Series in Probability and Statistics, 2012.
- [35] A. Cully, K. Chatzilygeroudis, F. Allocati, and J.-B. Mouret, “Limbo: A fast and flexible library for bayesian optimization,” *arXiv preprint arXiv:1611.07343*, 2016.
- [36] D. Lizotte, T. Wang, M. Bowling, D. Schuurmans, et al., “Gaussian process regression for optimization,” in *NIPS Workshop on Value of Information*, 2005.
- [37] J.-B. Mouret and J. Clune, “Illuminating search spaces by mapping elites,” *arXiv preprint arXiv:1504.04909*, 2015.
- [38] J. K. Pugh, L. B. Soros, and K. O. Stanley, “Quality diversity: A new frontier for evolutionary computation,” *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016.
- [39] M. Duarte, J. Gomes, S. M. Oliveira, and A. L. Christensen, “Evolution of repertoire-based control for robots with complex locomotor systems,” *IEEE Transactions on Evolutionary Computation*, 2017.
- [40] A. Gaier, A. Asteroth, and J.-B. Mouret, “Feature space modeling through surrogate illumination,” in *Proc. of GECCO*, 2017.
- [41] A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images,” in *Proc. of CVPR*, 2015.
- [42] V. Vassiliades, K. Chatzilygeroudis, and J.-B. Mouret, “Using Centroidal Voronoi Tessellations to Scale Up the Multi-dimensional Archive of Phenotypic Elites Algorithm,” *IEEE Trans. on Evolutionary Computation*, 2017.
- [43] J.-B. Mouret and S. Doncieux, “Sfresv2: Evolving in the Multi-Core World,” in *Proc. of CEC*, 2010.
- [44] M. Blum and M. A. Riedmiller, “Optimization of Gaussian process hyperparameters using Rprop,” in *Proc. of ESANN*, 2013.
- [45] R. A. Brooks, “Intelligence without representation,” *Artificial intelligence*, vol. 47, no. 1–3, pp. 139–159, 1991.
- [46] R. Pfeifer and J. Bongard, *How the body shapes the way we think: a new view of intelligence*. MIT press, 2006.